буде генерувати окремі ключі. В такому випадку адміністратору безпеки центру необхідно буде під час формування запису про робоче місце вказати, що ключ цього робочого місця може використовуватися відразу у декількох системах та вказати ролі, які буде виконувати цей ключ. Необхідно відзначити, що набір систем та ролей не є фіксованим, а може бути модифікований у центрі верхнього рівня. При інсталяції центрів нижчого рівня та генераторів система проконтролює правильність формування списків ролей та систем та виконає синхронізацію.

Сертифікати. Сертифікати центр зберігає у захищеній базі даних власного формату, що не може бути прочитана стандартними системами керування базами даних. Сертифікати по інформації, що входить до їх складу, відповідають стандарту X.509. База сертифікатів підписується та хешується, крім того, кожен відкритий сертифікат шифрується та хешується. У базі сертифікатів можуть зберігатися як дійсні сертифікати, так і нові, що ще тільки мають вступити у дію, та щойно видалені. Центр має змогу заблокувати роботу сертифікату, якщо виникла підозра на компрометацію відповідного секретного ключа.

## IX Система безпеки ЦУСК „Джерело”

Розглянемо тепер систему безпеки центру сертифікації. Для доступу до системи необхідно мати пароль. Крім того, навіть після входу у систему, основні дії можна виконати, тільки за допомогою носія з ключем сертифікації. Функції зміни структури системи контролюються адміністратором безпеки, який також має персональний ключ для роботи зі структурою системи. Сертифікат включає в себе поля лічильників цифрового підпису та шифрування, що захищає систему від можливості оновлення системи після виконання деяких операцій. Усі дії, починаючи від запуску системи і до завершення роботи програми заносяться у журнал операцій центра, де записуються такі дані: час, фамілія відповідального, найменування операції, над чим проводилась (якщо треба), статус виконання операції. Журнали та центр разом з іншими службовими файлами контролюються на цілісність.

## Висновки

Надаються основні терміни та поняття Законів України “Про електронний цифровий підпис” та “Про електронні документи та електронний документообіг” і задачі по створенню центрів сертифікації. Проводиться аналіз можливості використання центрів сертифікації Windows 2000 для цих центрів. Виконується аналіз архітектур центрів з урахуванням їх використання в розгалужених мережах. Описується центр сертифікації „ДЖЕРЕЛО”

Відповідно законам [1], [2] в Україні повинні бути створені Центральний засвідчу вальний орган, Центральний та відомчі Засвідчувальні центри, Акредитовані центри, та центри сертифікації ключів. Виконано аналіз можливості використання центрів сертифікації ключів Windows 2000 для цих центрів. Аналіз показав, що використання цих центрів неможливо. Визначена архітектура системи, яка відноситься до комбінованих систем. Ця система складається з декількох ієрархічних систем. Наведена структура, функції, можливості використання система „ДЖЕРЕЛО” в якості центрів сертифікації всіх рівнів відповідно [2] для банківських та корпоративних мереж.

**УДК 638.235.231**

# О ПРИМЕНЕНИИ КРИПТОГРАФИЧЕСКИХ СРЕДСТВ ОС WINDOWS ДЛЯ МАСКИРОВАНИЯ ИНФОРМАЦИИ

***Евгений Клименко***
*НИЦ “ТЕЗИС” НТУУ “КПИ”*

*Анотация:* **Рассматриваются вопросы применения криптографических средств, совместимых с ОС Windows, для защиты информации, циркулирующей в компьютерных системах.**
*Summary:* **Considered questions of the using the cryptographic facilities compatible with OS Windows.**
*Ключевые слова:* **Криптография, криптографические интерфейсы, ключи, энтропия, Crypto API, PGP SDK.**

## Введение

Задачи защиты информации, циркулирующей в компьютерных системах всегда актуальны. Прежде всего, это связано с возросшей популярностью т.н. «сильно распределенных» приложений, состоящих из

отдельных модулей, расположенных на различных компьютерах и взаимодействующих между собой с помощью сетевых протоколов. Задачи защиты информации не менее актуальны для приложений Интернет и клиентов электронной почты.

О росте актуальности защиты информации свидетельствует растущая популярность технических решений и научных работ, посвященных различным аспектам компьютерной безопасности. Примером такого положения может служить заявление руководства компании Pretty Good Privacy (PGP), специализирующейся на криптографических программных решениях, о свободном распространении пакета для разработки прикладных программ собственного изготовления PGP SDK, а также включение фирмой Microsoft в поставку программного продукта Internet Explorer версии 3.02 и операционной системы (ОС) Windows 95 OSR 2 базовых функций поддержки криптографии Win32 Crypto API 1.0. При этом следует упомянуть, что PGP SDK является кроссплатформенным пакетом, доступным как для пользователей Windows, так и для пользователей Linux и MacOS.

К сожалению, на сегодняшний день вопросы применения криптографических интерфейсов, как входящих в состав существующих ОС, так и поставляемых сторонними производителями, не достаточно полно отражены в литературе, издаваемой в Украине. Целью данной статьи является краткое описание возможностей указанных интерфейсов и демонстрация их практического применения на конкретних примерах.

## Описание криптографического интерфейса Win32 Crypto API

Среди ныне существующих криптографических интерфейсов наибольшей популярностью пользуется Microsoft Crypto API, входящий в состав операционных систем фирмы Microsoft. Своей популярностью он обязан, помимо широкой распространенности ОС семейства Windows, наличием качественной документации, возможностью расширения и усовершенствования со стороны сторонних разработчиков, а также удобством интеграции с существующими и новыми программными продуктами. В последних версиях ОС от Microsoft, базирующихся на технологии NT 5 (Windows 2000/XP/2003), помимо базового интерфейса Crypto API 1.0, поддерживающего базовые криптографические операции шифрования/дешифрования, цифровой подписи и обмена ключами, имеется также более совершенный интерфейс Crypto API 2.0. Кроме базовых криптографических преобразований он поддерживает также целый ряд функций, реализующих преобразования на более высоком уровне – поддержку сертификатов X.509, криптографических сообщений PKCS#7, защиты секретов DPAPI и др. Тем не менее, подавляющее большинство функций Crypto API 2.0 напрямую обращается к базовым функциям Crypto API 1.0 [2].

Все функции поддержки интерфейса Crypto API 1.0 содержатся в библиотеке advapi32.dll. За исключением нескольких сервисных функций (например CryptSetProvider) эти процедуры выполняют ряд вспомогательных операций и вызывают библиотеку в которой непосредственно реализованы соответствующие криптографические преобразования. Такие библиотеки называются криптопровайдерами (КП) (Cryptographic Service Provider, CSP). КП имеют стандартный набор функций, который состоит из 23 обязательных и 2 необязательных процедур.

Наиболее важным параметром каждого КП являются поддерживаемые криптографические алгоритмы для шифрования, хеширования, цифровой подписи и несимметричного обмена ключами.

Кроме различия в реализуемых алгоритмах КП отличаются способом физической организации ключевой базы. С точки зрения программирования способ физической организации ключевой базы значения не имеет. Однако он весьма важен с точки зрения эксплуатации и безопасности системы. Существующие КП Microsoft хранят свою ключевую базу на жестком диске (в Реестре или в файлах), а КП фирм Gemplus, Schlumberger и Infenion, которые включены в поставку Windows 2000, Windows XP и Windows ME — на смарт-картах.

Хотя способы физической организации ключевой базы (КБ) разных КП отличаются, логическая структура, определяемая самим интерфейсом, для всех одинакова. КБ представляется набором ключевых контейнеров. Каждый ключевой контейнер имеет имя, которое присваивается ему при создании, а затем используется для работы с ним. В ключевом контейнере сохраняется долговременная ключевая информация. В КП Microsoft долговременными являются ключевые пары цифровой подписи и несимметричного обмена ключами, которые также генерируются функциями CryptoAPI.

Рассмотрим подробно, каким образом функции CryptoAPI из библиотеки advapi32.dll вызывают запрошенный пользователем КП и какие операции осуществляются, прежде чем вызывается конкретный КП, характеризующийся собственным именем и типом. Его имя представляет собой строку по которой система распознает КП. Базовый КП Microsoft назван «Microsoft Base Cryptographic Provider». Тип КП обозначается целым числом без знака (в нотации языка C — DWORD), значение которого идентифицирует набор поддерживаемых алгоритмов цифровой подписи и несимметричного обмена ключей. Уже

упомянутый КП «Microsoft Base Cryptographic Provider» имеет тип 1 (в файле WinCrypt.h определена константа PROV_RSA_FULL). Этот тип КП реализует в качестве алгоритмов цифровой подписи и несимметричного обмена ключей стандарт RSA. Другой базовый КП Microsoft - «Microsoft Base DSS and Diffie-Hellman Cryptographic Provider» - имеет тип 13 (в файле WinCrypt.h определена константа PROV_DSS_DH). Данный тип КП реализует в качестве алгоритма цифровой подписи стандарт DSS, а для реализации ключевого обмена использует алгоритм ElGamal. В системе могут существовать несколько различных КП одного и того же типа. КП фирм Gemplus, Schlumberger и Infenion имеют тот же тип, что и КП «Microsoft Base Cryptographic Provider», и реализуют цифровую подпись и ключевой обмен по алгоритму RSA.

Для работы с набором КП в системном реестре содержится список имен всех КП. С каждым именем связан тип КП и имя библиотеки, реализующей весь набор функций. Кроме того, в системе содержится информация о том, какой КП применять, если пользователь при вызове не определил конкретное имя, а задал только тип необходимого ему КП. Такие КП называются КП, используемыми по умолчанию (default cryptographic service provider) для данного типа. Для типа 1 КП по умолчанию является «Microsoft Base Cryptographic Provider», а для типа 13 - «Microsoft Base DSS and Diffie-Hellman Cryptographic Provider». Для определения имени КП по умолчанию применяется API-функция CryptGetDefaultProvider, а для изменения этого параметра – API-функция CryptSetProvider или CryptSetProviderEx. Из описания этих функций следует, что КП по умолчанию задается как для текущего пользователя (current user), так и для системы в целом (local computer). Другие КП задаются для всех типов КП, установленных при инсталяции ОС, и сохраняются в ключе реестра HKEY_LOCAL_MACHINE\\Software\\Microsoft\\Cryptography\\Defaults\\Provider Types\\Type. Имена всех установленных в системе КП расположены в системном реестре по пути HKEY_LOCAL_MACHINE\\Software\\Microsoft\\Cryptography\\Defaults\\Provider (рис. 1).

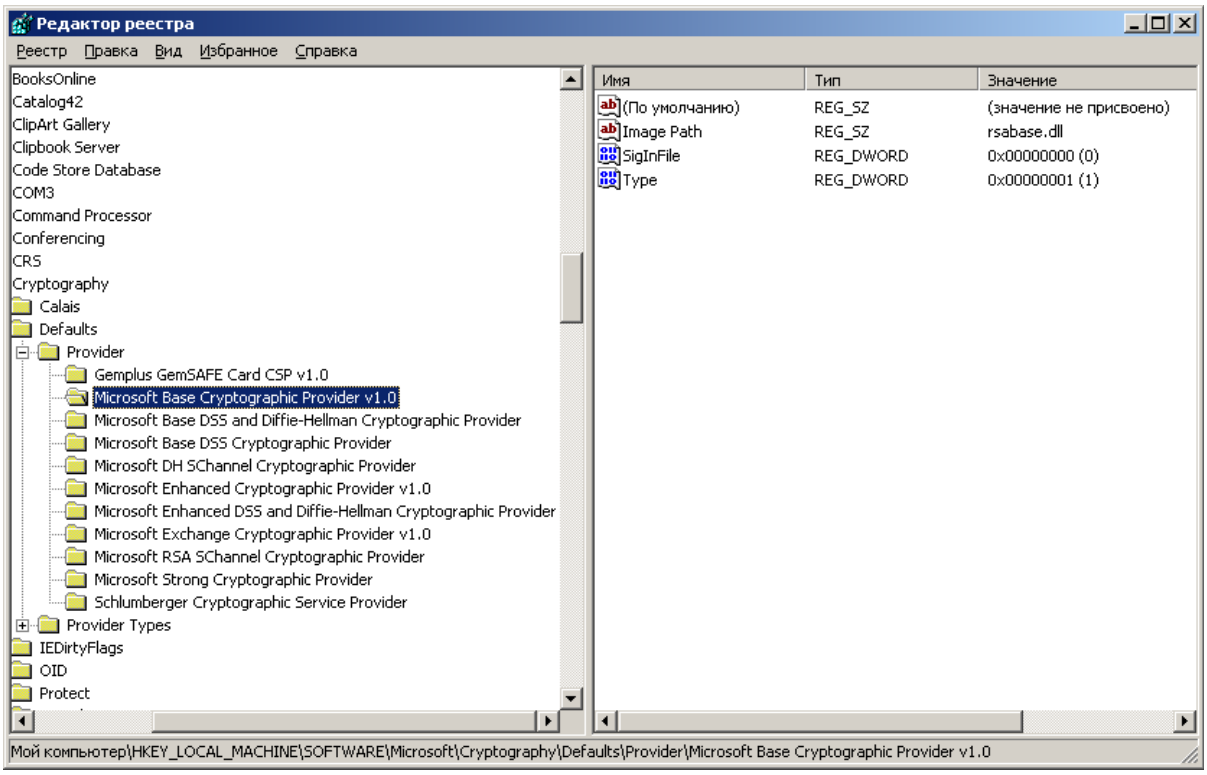


**Рисунок 1 – ключ реестра, где хранятся имена криптопровайдеров, установленных в системе**

В дальнейшем можно изменить эти параметры или назначить КП по умолчанию только для текущего пользователя. Параметры для текущего пользователя имеют приоритет перед общесистемными и сохраняются в ключе реестра HKEY_CURRENT_USER\\Software\\Microsoft\\Cryptography. Если параметры для текущего пользователя в явном виде отсутствуют, то применяются общесистемные.

## Программа для маскирования на базе интерфейса Win32 Crypto API

Программа cryptapi, разработанная автором, демонстрирует применение функций интерфейса Crypto API 2.0 для маскирования текстовых сообщений с помощью симметричного алгоритма 3DES.

Программа представляет собой консольное приложение Win32, которое, в отличии от графических приложений, выводит информацию в текстовом виде. Для вывода сообщений используется кодировка UNICODE UTF16, которая представляет буквенно-цифровые символы в формате т. н. «широких символов» (wide char) размером 16 бит. Основным преимуществом данной кодировки по сравнению со стандартной кодировкой ANSI является поддержка региональных наборов символов (кириллица, арабский, китайский и др.) без необходимости поддержки кодовых страниц. Поскольку поддержка интерфейса Crypto API 2.0, ровно как и кодировки UNICODE, присутствует только в системах семейства Windows NT, то в системах Windows 9x/ME данная программа выведет на консоль сообщение об ошибке и завершит выполнение.

Запуск программы осуществляется из командной строки Windows со следующим набором параметров:
*cryptapi -e [имя файла с исходным текстом] [имя файла для шифротекста]* – для шифрования;
*cryptapi -d [имя файла с шифротекстом] [имя файла для восстановленного исходного текста]* – для дешифрования.

Конструктивно программа разделена на три блока – инициализации, шифрования и дешифрования.

Блок инициализации оформлен в виде функции стандартной библиотеки языка C (C Runtime Library) для приложений с поддержкой UNICODE, которая имеет следующий прототип:

```
void _cdecl wmain(int argc, wchar_t* argv[]);
```

Данная функция начинает работу с анализа типа ОС. Если программа запущена под управлением системы, не поддерживающей технологию NT, выводится сообщение об ошибке и работа программы завершается.

На следующем шаге программа анализирует параметры командной строки. Если параметры переданы корректно, то вызываются соответствующие функции в зависимости от затребованной операции — в противном случае работа завершается с ошибкой.

Блок шифрования представляет собой функцию со следующим прототипом:

```
void _cdecl EncryptFile(wchar_t* source_file, wchar_t* dest_file);
```

Функция выполняет такую последовательность действий.

1. Инициализация КП вызовом функции **CryptAcquireContext**. Если для данной учетной записи отсутствует контейнер ключей, то он создается повторным вызовом **CryptAcquireContext**.
2. Открытие файла с исходным текстом.
3. Создание файла для размещения шифротекста.
4. Запрос у криптопровайдера ключа для обмена (для криптопровайдера «Microsoft Strong Cryptographic Provider» ключ для обмена представляет собой открытый ключ стандарта RSA). Если готовый ключ отсутствует, то генерируется новый ключ и сохраняется в базе данных КП.
5. Генерируется сеансный ключ для алгоритма 3DES.
6. В память приложения в двоичном формате передается сеансный ключ 3DES, зашифрованный ключом для обмена RSA (key blob format).
7. В начало файла для шифротекста записывается сначала длина сеансного ключа, а затем сам сеансный ключ.
8. Данные шифруются отдельными блоками и записываются в файл.
9. Выполняется очистка ресурсов, после чего функция возвращает управление.

Блок дешифрования представляет собой функцию с таким прототипом:

```
void _cdecl DecryptFile(wchar_t* source_file, wchar_t* dest_file);
```

Функция выполняет такие операции.

1. Инициализация криптопровайдера.
2. Открытие файла с шифротекстом.
3. Создание файла для размещения восстановленного исходного текста.
4. Считывание параметров сеансного ключа, записанного в начало файла с шифротекстом.
5. Запись полученных параметров в буфер и отправление криптопровайдеру.
6. Расшифрование блоков данных и запись в файл для восстановленного исходного сообщения.
7. Очистка ресурсов.

После этого функция возвращает управление основной функции.

Сборка и отладка программы производилась с помощью интегрированной среды разработки программ Microsoft Visual Studio версии 6.0. Для сборки программы в других известных средах разработки приложений Win32 – Borland Delphi и Borland C++ Builder – необходимы лишь незначительные изменения исходного кода.

## Описание криптографического интерфейса PGP SDK

Помимо Microsoft Crypto API, существует ряд популярных криптографических интерфейсов. Одним из них является криптоинтерфейс PGP фирмы Phil's Pretty Good Software. В состав данного криптоинтерфейса входит комплект разработки программ (PGP Software Development Kit или сокращенно PGP SDK), позволяющий разработчикам программного обеспечения интегрировать криптографические технологии PGP в собственные программы. Криптоинтерфейс PGP позволяет выполнять три базовые криптографические операции, а именно - управление ключами, кодирование и аутентификацию.

Управление ключами реализовано в виде следующих операций:

- ✓ создание ключей и/или добавление в базу данных;
- ✓ удаление из базы данных;
- ✓ поиск ключей по различным параметрам;
- ✓ проверка корректности контейнеров ключей на диске или в памяти;
- ✓ проверка и/или установка значений свойств ключей;
- ✓ создание, удаление и модификация ключевых контейнеров и групп ключей.

Кодирование реализовано в виде следующих операций:

- ✓ шифрование данных или файлов;
- ✓ дешифрование данных или файлов.

Аутентификация реализована в виде следующих операций:

- ✓ цифровая подпись сообщений или файлов данных;
- ✓ проверка подлинности сообщений или файлов данных.

Прочие функции реализуют модули генерации псевдослучайных чисел, манипуляции с большими числами, а также сервисные функции и функции для доступа к серверу ключей.

Криптоинтерфейс PGP SDK поддерживает также функции графического интерфейса пользователя (Graphic User Interface - GUI) при помощи которых можно выполнять следующие операции:

- ✓ выбор ключей получателя;
- ✓ ввод пропускной фразы;
- ✓ ввод пропускной фразы с подтверждением;
- ✓ ввод пропускной фразы для отдельного ключа;
- ✓ ввод пропускной фразы для проверки цифровой подписи.

Следует отметить, что функции и структуры данных PGP SDK намного сложнее Crypto API. Однако такое положение вполне оправдано, так как предоставляет разработчику широкий набор возможностей и существенную гибкость при выполнении криптографических операций. Также следует отметить, что успешное использование PGP SDK предполагает наличие более глубоких знаний криптографии, в то время как для работы с Crypto API вполне достаточно базовых.

## Программа для маскирования на базе интерфейса PGP SDK

Программа pgpencode, разработанная автором, демонстрирует применение API-функций интерфейса PGP SDK 1.7.2 для маскирования текстовых сообщений с помощью симметричного алгоритма 3DES.

Как и пример с Crypto API, данная программа представляет собой консольное приложение Win32 с поддержкой UNICODE. Исходя из этого, корректная работа программы возможна только под управлением операционной системы с технологией NT.

Запуск программы осуществляется из командной строки Windows со следующим набором параметров:
*pgpencode -e [имя файла с исходным текстом] [имя файла для шифротекста] [имя файла открытого ключа]* – для шифрования;
*pgpencode -d [имя файла с шифротекстом] [имя файла для восстановленного исходного текста] [имя файла секретного ключа]* – для дешифрования;
*pgpencode -c [имя файла открытого ключа] [имя файла секретного ключа]* – для создания ключевой пары.

Конструктивно программа разделена на четыре блока – инициализации, генерации ключевой пары, шифрования и дешифрования.

Блок инициализации оформлен в виде функции стандартной библиотеки языка C (C Runtime Library) для приложений с поддержкой UNICODE, которая имеет следующий прототип:

```
void _cdecl wmain(int argc, wchar_t* argv[]);
```

Реализация функции **wmain** в основном выполняет те же действия, что и реализация для примера с Crypto API.

Блок генерации ключевой пары представляет собой функцию со следующим прототипом:

```
bool CreateNewKeyPair(char* szPubRingFile, char* szSecRingFile, char*
szUserID);
```

Функция выполняет такую последовательность действий:

1. Инициализация PGP SDK.
2. Инициализация поддержки пользовательского интерфейса.
3. Создание контекста PGP.
4. Создание файла для размещения открытого ключа в формате файловой спецификации PGP.
5. Создание файла для размещения секретного ключа в формате файловой спецификации PGP.
6. Создание хранилища для ключевой пары (key pair ring).
7. Создание внутри данного хранилища пустого набора ключей (key set).
8. Вычисление показателя энтропии для набора случайных чисел, необходимого для создания ключевой пары стандарта RSA с длиной ключа 1024 бит. В отличии от Crypto API, где место для хранения и длина ключей определяется самим интерфейсом, в PGP SDK разработчик имеет возможность задавать указанные параметры по своему усмотрению. Кроме того, если в Crypto API требовалось явное определение назначения ключей (для шифрования, цифровой подписи или обмена), то PGP SDK таких жестких условий не ставит. Отметим также, что по оценкам экспертов для обеспечения гарантированной стойкости шифрограммы минимальная длина ключа RSA должна составлять не менее 1024 бит (рекомендуется 2048).
9. Инициализация и отображение диалога случайного ввода. Пользователю предлагается перемещать указатель мыши в пределах рабочей области диалога. Необходимый для генерации ключевой пары набор случайных чисел формируется путем считывания координат перемещений мыши (рис. 2).
10. Формирование списка параметров для генерации ключевой пары. В отличие от Crypto API, PGP SDK позволяет задавать обширный набор опций для генерации ключей – метку времени (time stamp), имя пользователя, пропускная фраза, срок действия ключа (expiration) и др.
11. Генерация ключевой пары.
12. Выдача информации о ключевой паре.
13. Добавление сгенерированной ключевой пары в набор ключей (key set).
12. Регистрация изменений в хранилище ключей.
15. Освобождение выделенных ресурсов.

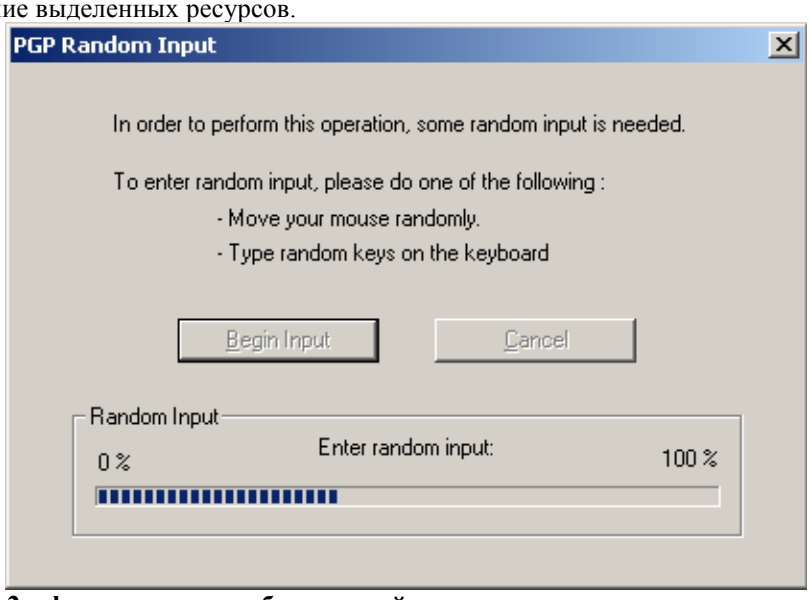


**Рисунок 2 – формирование набора случайных чисел для генерации ключевой пары**

Блок шифрования представляет собой функцию со следующим прототипом:

```
bool EncryptData(char* szInputFile, char* szOutputFile, char*
szPubRingFile, char* szUserID);
```

Функция выполняет такую последовательность действий.

1. Инициализация PGP SDK.
2. Создание контекста PGP.
3. Создание файловой спецификации для файла с открытым ключом.
4. Импорт ключей из файла в хранилище ключей.

5. Поиск нужного ключа по критерию. В приведенном примере критерием является имя пользователя.
6. Открытие файла с исходным текстом в формате файловой спецификации PGP.
7. Создание файла для размещения шифротекста в формате файловой спецификации PGP.
8. Формирование опций для шифрования. На основании опций формируется сеансный ключ 3DES, закодированный открытым ключом RSA, взятым из файла.
9. Шифрование данных и запись результатов в файл.
10. Очистка выделенных ресурсов.

Блок дешифрования представляет собой функцию со следующим прототипом.

```
bool DecryptData(char* szInputFile, char* szOutputFile, char* szSecRingFile, char* szUserID);
```

Функция выполняет такую последовательность действий.
1. Инициализация PGP SDK.
2. Создание контекста PGP.
3. Создание файловой спецификации для файла с секретным ключом.
4. Импорт ключей из файла в хранилище ключей.
5. Поиск нужного ключа по критерию. В приведенном примере критерием является имя пользователя.
6. Открытие файла с шифротекстом в формате файловой спецификации PGP.
7. Создание файла для размещения восстановленного текста в формате файловой спецификации PGP.
8. Формирование опций для дешифрования. На основании опций формируется сеансный ключ 3DES, закодированный секретным ключом RSA, взятым из файла.
9. Дешифрование данных и запись результатов в файл.
10. Очистка выделенных ресурсов.

Сборка и отладка программы производилась с помощью интегрированной среды разработки программ Microsoft Visual Studio версии 6.0. В отличии от программ с поддержкой Crypto API, перенос программ с поддержкой PGP SDK на платформы Borland Delphi и Borland C++ Builder является нетривиальной задачей. Тем не менее, на сегодняшний день имеется ряд решений, доступных через Интернет.

Для сборки приведенной программы необходимо скачать комплект PGP SDK на одном из следующих Интернет-порталов:

| |
|---|
| http://www.pgpi.org/products/sdk/c++/pgpsdk/ |
| ftp://ftp.hacktic.nl/pub/crypto/pgp/pgpsdk/ |
| ftp://ftp.no.pgpi.org/pub/pgp/sdk/ |

В PGP SDK входят заглавные файлы, библиотеки, документация с описанием функций и тестовые примеры. Установка комплекта PGP SDK заключается в копировании этого архива, распаковки его в отдельный каталог и прописывании путей к каталогам *Headers (Include files)* и *Libraries\DLL\Release (Library files)* в настройках Visual Studio *(Tools - Options - Projects - VC++ Directories).*

В заключении необходимо скопировать отладочные версии dll-файлов (PGP_SDK.dll, PGPsdkUI.dll и PGPsdkNL.dll**)** в директорию проекта.

## Заключение

На сегодняшний день на рынке имеется достаточно богатый набор средств компьютерной криптографии, ориентированных как на использование штатних криптографических средств ОС Windows, так и криптографических пакетов сторонних производителей, позволяющих выполнять криптографические преобразования над данными любой структуры и сложности.

Применение, начиная с Windows 2000, технологии цифровой подписи системных драйверов существенно повышает стабильность системы и практически предотвращает проникновение вирусов, работающих на уровне системы или на аппаратном уровне и представляющих наибольшую опасность для ОС. Кроме того, начиная с Windows XP, в Crypto API появилась поддержка нового криптолагоритма повышенной стойкости AES.

Программы, разработанные автором, демонстрируют набор операций, необходимых для осуществления маскирования текстовой информации средствами Win32 Crypto API и PGP SDK, а именно:

- ✓ работа с хранилищами и наборами ключей;
- ✓ генерация ключевых пар;
- ✓ работа с псевдослучайными числами;
- ✓ поиск и сортировка ключей;
- ✓ шифрование;

- ✓ дешифрование.

Более того, данные программы могут использоваться в качестве примера для разработки реальных криптографических приложений.

*Література:* ***1.*** *Ховард М., Лебланк Д. Защищенный код/Пер. с англ. – М.: Издательско-торговый дом «Русская редакция», 2003. – 704 стр.: ил.* ***2.*** *Щербаков А. Ю., Домашев А. В. Прикладная криптография. Использование и синтез криптографических интерфейсов. – М.: Издательско-торговый дом «Русская редакция», 2003. – 416 с.: ил.* ***3.*** *Platform SDK: Security. Cryptography - Microsoft Platform SDK, February 2001 Edition.* ***4.*** *PGP Software Development Kit, Reference Guide. Version 1.7.1 Int. - Copyright © 1990-1999 Network Associates, Inc. and its Affiliated Companies. All Rights Reserved. – www.pgp.org* ***5.*** *PGP Software Development Kit, User's Guide. Version 1.7.1 Int. - Copyright © 1990-1999 Network Associates, Inc. and its Affiliated Companies. All Rights Reserved. – www.pgp.org* ***6.*** *Хеширование, шифрование и цифровая подпись с использованием CryptoAPI и .NET – www.rsdn.ru*

**УДК 681.3.06**

# СТОЙКИЙ К КОЛЛИЗИЯМ АЛГОРИТМ WHIRLPOOL

***Геннадий Халимов, Евгений Котух***

*Харьковский национальный университет радиоэлектроники*

*Аннотация:* **Рассмотрены требования к хеш функциям устойчивым к коллизиям, алгоритм Whirlpool и аспекты безопасности криптопримитива .**

*Suumary:* **Requirements for collision-resistant hash functions, Whirlpool algorithm and aspects of crypto primitive security are considered.**

*Ключевые слова:* **алгоритм Whirlpool, бесключевая хеш-функция.**

## I Введение

Разнообразие подходов к решению проблем аутентификации обусловлено требованиями к системе защиты информации. Среди основных подходов к построению MAC кодов, широкое распространение получили коды на основе бесключевых хэш-функций (MDC-коды), с применением универсальных хэш-функций, а также MAC коды, использующие блочные шифры.

Применение бесключевых хеш-функций является эффективным методом для построения MAC кодов, так как обеспечивает высокую стойкость к коллизиям и скорость вычислений. В проекте Nessie такие MAC коды представлены как HMAC коды и их конструкция описана в RFC 2104 [1]:

Безопасность HMAC алгоритма определяется секретностью бесключевой хеш-функции. В проекте NESSIE лучшим в категории «Устойчивая к коллизиям хэш-функция» был признан алгоритм Whirlpool. Алгоритм Whirlpool вычисляет 512-битный хеш-код с использованием в качестве функции сжатия блочного шифра, котрый является модификацией алгоритма Rijndael.

С этой целью в разделе II рассмотрены требования к хэш-функциям, устойчивым к коллизиям. В разделе 2 приводится описание алгоритма Whirlpool, оценки и сравнение хэш-функций. В разделе 3 изучены аспекты безопасности криптопримитива Whirlpool.

## II Требования безопасности к хэш-функциям, устойчивым к коллизиям

Конструктивными элементами HMAC кодов являются хеш-функции, функции сжатия и итерационные хеш-функции. Определения отмеченных хеш-функций и их криптографических свойств приведем в изложении Блэка и Рогавэя [2].

**Определение 1. Хеш-функцией** называется функция отображения $h: D \rightarrow R$ , где область значений $D = \{0,1\}^*$, а $R = \{0,1\}^n$ для некоторого $n \geq 1$.

**Определение 2. Функцией сжатия** называется функция отображения $f: D \rightarrow R$, где $D = \{0, 1\}^a \times \{0,1\}^b$ и $R = \{0,1\}^n$ для некоторых a, b, $n \geq 1$ и $a + b \geq n$.

**Определение 3. Итерационной хеш-функцией от** функции сжатия $f: (\{0,1\}^n \times \{0, 1\}^b) \rightarrow \{0,1\}^n$ является хеш-функция $h: (\{0, 1\}^b)^* \rightarrow \{0,1\}^n$ определенная $h(X_1 \ldots X_t) = H_t$, где $H_i = f(H_{i-1}, X_i)$ при $1 \leq i \leq t$ ($H_0 = IV$).

Определяющими требованиями к хеш-функциям являются их стойкость к вычислению прообраза, второго прообраза, а также стойкости к коллизиям.

**Определение 4 (Стойкость к вычислению прообраза) [3].** Хеш-функция $h: \{0,1\}^* \rightarrow R$ является